Joint multiple dictionary learning for tensor sparse coding

Joint Multiple Dictionary Learning for Tensor Sparse Coding

Yifan Fu, Junbin Gao, Yanfeng Sun and Xia Hong

Abstract— **Traditional dictionary learning algorithms are used for finding a sparse representation on high dimensional data by transforming samples into a one-dimensional (1D) vector. This 1D model loses the inherent spatial structure property of data. An alternative solution is to employ Tensor Decomposition for dictionary learning on their original structural form —a tensor— by learning multiple dictionaries along each mode and the corresponding sparse representation in respect to the Kronecker product of these dictionaries. To learn tensor dictionaries along each mode, all the existing methods update each dictionary iteratively in an alternating manner. Because atoms from each mode dictionary jointly make contributions to the sparsity of tensor, existing works ignore atoms correlations between different mode dictionaries by treating each mode dictionary independently. In this paper, we propose a joint multiple dictionary learning method for tensor sparse coding, which explores atom correlations for sparse representation and updates multiple atoms from each mode dictionary simultaneously. In this algorithm, the Frequent-Pattern Tree (FP-tree) mining algorithm is employed to exploit frequent atom patterns in the sparse representation. Inspired by the idea of K-SVD, we develop a new dictionary update method that jointly updates elements in each pattern. Experimental results demonstrate our method outperforms other tensor based dictionary learning algorithms.**

I. Introduction

SPARSE CODING (SC) has been widely applied in numerous signal processing tasks, such as imaging denoising [13], texture synthesis [15] and image classification [18]. Sparse modeling aims at learning a dictionary so that each sample can be represented by a few atoms of the learned dictionary. Several algorithms have been developed for this task, e.g. the K-SVD [1] and the method of optimal directions (MOD) [6]. However, when dealing with multi-dimensional signals , all the previous sparse models reshape each input signal into a 1D vector. This kind of reshaping breaks the local correlation inherent inside the signal. Thus, it becomes highly desirable to develop algorithms which are able to fully make use of the local correlation inside high-dimensional data, such as tensor data.

Tensor decomposition [12] (TD) has attracted attention for processing multi-dimensional data. PARAFAC [12] and TUCKER [12] decompositions are two classical algorithms. PARAFAC decomposes a tensor as a sum of k rank-1 tensor while TUCKER factorizes a tensor into a set of matrices and one small core tensor. However, these methods do not explicitly enforce sparsity constraint in the low dimensional representation.

Recently, researchers resort to combining SC with TD by introducing additional constraints to the models with the aim of learning sparse representations of tensors. Both non-negativity and sparsity have been used in two classical decompositions to accomplish the goal. Both non-negative versions of PARAFAC and TUCKER decompositions with multiplicative updates have been proposed in [9], [2] and [11], [10]. In TUCKER model, the sparsity over the core tensor is achieved by smoothing matrices along each mode [10], l_1 norm penalization [14] or a tensor dictionary learning algorithm [19].

In the case of dictionary learning models, which is the focus of our discussion in this paper, Caiafa and Cichocki [4] open the discussion of sparse representation of tensor data using Kronecker bases in which the size of the core tensor is much higher than the input tensor. Two models are proposed in this paper: (1) Kronecker- Orthogonal Matching Pursuit (OMP) algorithm for multiway sparsity in which the sparse non-zero coefficients could be distributed randomly in the core tensor; (2) N-way Block OMP (N-BOMP) for multiway block sparsity in which the non-zero entries of the core tensor form blockwise structure. The similar ideas are used in 2 dimensional dictionaries for image processing [17]. In [19], a tensor dictionary model based on sparse TUCKER decomposition is proposed, in which sparse constraint over the core tensor is achieved by N-OMP, and the n-mode dictionary is learnt in an alternative minimization manner using gradient descent.

However, all existing dictionary learning algorithms update each mode dictionary by fixing all the other mode dictionaries iteratively in an alternating manner. The main issue is that the atoms from each mode dictionary jointly make contributions to the presentation of tensors, while current dictionary learning solutions ignore atoms correlations between different mode dictionaries. To this end, we propose a joint multiple dictionary learning method for tensor sparse coding (TSC-JMDL), which explores atom correlations for sparse representation and updates multiple atoms from each mode dictionary simultaneously. Our main contributions are as follows:

- Unlike pervious tensor dictionary learning algorithms update all the non-zero atom entries for each mode, our model employs FP-tree to find the frequent atom patterns formed by a sequence of non-zero atom entries in the sparse core tensor, and updates only frequent patterns by a tensor extended version of K-SVD algorithm.
- Existing tensor dictionary learning models update each

Yifan Fu and Junbin Gao are with School of Computing and mathematics, Charles Sturt University, Bathurst, NSW 2795, Australia. (email: {yfu, jbgao}@csu.edu.au).

Yanfeng Sun is with Beijing Municipal Key Lab of Multimedia and Intelligent Software Technology, Beijing University of Technology, Beijing 100124, China. (email: yfsun@bjut.edu.cn).

Xia Hong is with School of Systems Engineering, University of Reading, Reading, RG6 6AY, UK. (email: x.hong@reading.ac.uk).

This work is supported by the Australian Research Council (ARC) through Discovery Project Grant DP130100364.

mode dictionary independently, without considering atom correlations for sparse representation. In contrast, our model jointly updates all the elements in a frequent atom pattern simultaneously.

II. Notations and Problem Formulation

A. Definition and Notations

A *tensor* is a multidimensional array. The *order* of a tensor is the number of dimensions, also known as ways or modes. For example, $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ is an N-way tensor, where $I_n (1 \leq n \leq N)$ are the dimensions of each mode. The element indexed by $(i_1, i_2, \ldots, i_N)$ in an N-way tensor is denoted by $y_{i_1,i_2,\ldots,i_N}$. In particular, a vector (1-way tensor) is denoted by a boldface lower-case letter, i.e. $\mathbf{y} \in \mathbb{R}^I$ and a matrix (2-way tensor) is denoted by a bold uppercase letter, i.e. $\mathbf{Y} \in \mathbb{R}^{I \times M}$. The i-th entry of a vector $\mathbf{y}$ is denoted as y_i, and the element at (i, j) of a matrix $\mathbf{Y}$ is denoted as y_{ij}. Thereafter, we will introduce some tensor fundamentals and definitions.

Definition 1 (Tensor Matricization): Matricizaion is the operation of rearranging the entries of a tensor so that it can be represented as a matrix. Let $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ be a tensor of order-N, the mode-n matricization of $\mathcal{X}$ reorders the mode-n vectors to be columns of the resulting matrix, denoted by $\mathbf{X}_{(n)} \in \mathbb{R}^{I_n \times (I_{n+1} I_{n+2} \ldots I_N I_1 I_2 \ldots I_{n-1})}$.

Definition 2 (Rank-One Tensor): An N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ is rank one if it can be equal to the outer product of N vectors:

$$\mathcal{X} = \mathbf{a}^{(1)} \circ \mathbf{a}^{(2)} \circ \ldots \circ \mathbf{a}^{(N)}. \tag{1}$$

Elementwise, we have

$$x_{i_1 i_2 \ldots i_N} = a_{i_1}^{(1)} a_{i_2}^{(2)} \ldots a_{i_N}^{(N)} \quad \text{for all} \quad 1 \leq i_n \leq I_n. \tag{2}$$

Definition 3 (Kronecker Product): The Kronecker product of matrices $\mathbf{A} \in \mathbb{R}^{I \times J}$ and $\mathbf{B} \in \mathbb{R}^{P \times L}$, denoted by $\mathbf{A} \otimes \mathbf{B}$, is a matrix of size $(IP) \times (JL)$ defined by

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & a_{12}\mathbf{B} & \cdots & a_{1J}\mathbf{B} \\ a_{21}\mathbf{B} & a_{22}\mathbf{B} & \cdots & a_{2J}\mathbf{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}\mathbf{B} & a_{I2}\mathbf{B} & \cdots & a_{IJ}\mathbf{B} \end{bmatrix} \tag{3}$$

Definition 4 (The n-mode Product): The n-mode product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ by a matrix $\mathbf{U} \in \mathbb{R}^{J \times I_n}$, denoted as $\mathcal{X} \times_n \mathbf{U}$, is a tensor with entries:

$$(\mathcal{X} \times_n \mathbf{U})_{i_1,\ldots,i_{n-1},j,i_{n+1},\ldots,i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \ldots i_N} u_{j i_n} \tag{4}$$

The n-mode product is also denoted by each mode-n vector multiplied by the matrix $\mathbf{U}$. Thus, it can be expressed in terms of tensor matricization as well:

$$\mathcal{Y} = \mathcal{X} \times_n \mathbf{U} \qquad \Leftrightarrow \qquad \mathbf{Y}_{(n)} = \mathbf{U}\mathbf{X}_{(n)} \tag{5}$$

Definition 5 (Tucker Decomposition): Given an order-N tensor $\mathcal{Y}$, its Tucker decomposition is an approximated tensor defined by,

$$\begin{aligned} \hat{\mathcal{Y}} &\equiv [\![\mathcal{X}; \mathbf{U}_1, ..., \mathbf{U}_N]\!] = \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \ldots \times_N \mathbf{U}_N \\ &= \sum_{i_1=1}^{M_1} \sum_{i_2=1}^{M_2} \cdots \sum_{i_N=1}^{M_N} x_{i_1 i_2 \ldots i_N} \mathbf{u}_{i_1} \circ \mathbf{u}_{i_2} \ldots \circ \mathbf{u}_{i_N} \end{aligned} \tag{6}$$

where $\mathcal{X} \in \mathbb{R}^{M_1 \times M_2 \times \ldots \ M_N}$ is called a core tensor, $\mathbf{U}_i \in \mathbb{R}^{I_i \times M_i} (1 \leq i \leq N)$ are the factor matrices and the symbol $\circ$ represents the vector outer product.

Definition 6 (CP Decomposition): Given an N-way tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \ldots \times I_N}$, the CP tensor decomposition factorizes a tensor into a sum of component rank-one tensors, which is defined as

$$\hat{\mathcal{X}} \equiv \sum_{i=1}^{M} \mathbf{u}_i^{(1)} \circ \mathbf{u}_i^{(2)} \cdots \circ \mathbf{u}_i^{(N)} \tag{7}$$

where M is a positive integer and $\mathbf{u}_i^{(j)} \in \mathbb{R}^{I_j}$ for $i = 1, \ldots, M$.

Definition 7 (Multiway block-sparsity): A tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times \ldots \times I_N}$ is $(S_1, S_2, \ldots, S_N)$-block sparse with respect to the factors $\mathbf{U}_n \in \mathbb{R}^{I_n \times M_n} (n = 1, 2, \ldots, N)$ if it admits a TUCKER representation based only on few S_n selected columns of each factor ($S_n \leq M_n$), i.e. if $\mathbf{i}_n = [i_n^1, i_n^2, \ldots, i_n^{S_n}]$ denotes a substet of indices for mode $n (n = 1, 2, \ldots, N)$, then

$$\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \ldots \times_N \mathbf{U}_N \tag{8}$$

with $x_{i_1 i_2 \cdots i_N} = 0 \ \forall (i_1 i_2 \cdots i_N) \notin \mathbf{i}_1 \times \mathbf{i}_2 \times \cdots \times \mathbf{i}_N$

B. Problem Formulation

Given a set of M N-order tensors, denoted by $\mathcal{Y} = \{\mathcal{Y}_1, \mathcal{Y}_2, \ldots, \mathcal{Y}_M\}$, we aim to find sparse representation for each N-order tensor $\mathcal{Y}_m (1 \leq m \leq M)$ with regards to the Kronecker product of multiple dictionaries $\mathbf{U}_n$ for $1 \leq n \leq N$. Each $\mathbf{U}_n$ is a dictionary along a particular direction of structure. To learn multiple dictionaries, we consider TUCKER decomposition on $\mathcal{Y}_m$. Thus the problem of multiple dictionary learning is formulated as

$$\min_{\mathcal{X}_m \text{sparse}, \mathbf{U}_n} \sum_{m=1}^{M} \|\mathcal{Y}_m - \mathcal{X}_m \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N\| \tag{9}$$

where $\mathcal{X}_m$ is the sparse representation tensor for $\mathcal{Y}_m$, and the factor matrix $\mathbf{U}_n$ is the dictionary at mode-n. In this paper, we suppose $\mathcal{X}_m$ is multiple block sparse as described in [4].

Naturally, we can regard $\mathcal{Y}$ an $(N+1)$-order tensor by stacking all the given tensors along the $(N+1)$-mode. Similarly $\mathcal{X}$ denotes the $(N+1)$-order tensor consisting of $\{\mathcal{X}_1, \mathcal{X}_2, \ldots, \mathcal{X}_M\}$. Then model (9) can be equivalently written as

$$\min_{\mathcal{X}_m \text{sparse}, \mathbf{U}_n} \|\mathcal{Y} - \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N \times_{N+1} \mathbf{I}\| \tag{10}$$

where $\mathbf{I}$ is the identity matrix of order M.

III. Joint Multiple Dictionary Learning Model for Tensor Sparse Coding

Similar to standard dictionary learning algorithms, the proposed tensor dictionary learning algorithm is a two-stage iterative process: sparse coding and dictionary update. In this paper, we employ an iterative algorithm called the Block Coordinate Descent (BCD) [3] to solve the optimization problem (10) by fixing all the other model variables to solve one variable at a time alternatively. Firstly, the dictionaries $\mathbf{U}_n$ are fixed , the sparse coefficients $\mathcal{X}_m$ can be obtained by solving M independent sparse representation subproblems. That is, for each $m = 1, \ldots, M$, $\mathcal{X}_m$ is obtained by minimizing

$$\min_{\mathcal{X}_m \text{sparse}} \|\mathcal{Y}_m - \mathcal{X}_m \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N\| \qquad (11)$$

with the N-BOMP algorithm.

Then in the second stage, given the sparse representations $\mathcal{X}_m$, the dictionaries (factors) corresponding to each mode are updated jointly using an atom pattern based dictionary update strategy, by solving the following problem

$$\min_{\mathbf{U}_n} \|\mathcal{Y} - \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N \times_{N+1} \mathbf{I}\| \qquad (12)$$

where $1 \leq n \leq N$.

A. N-BOMP Algorithm

OMP iteratively refines a sparse representation by successively identifying one component at a time that yields the greatest improvement in quality until a desired sparsity level is reached or the approximation error is below a predefined threshold. A tensor extended version of OMP for multiway block-sparsity is motivated by the fact that, in the real world, the non-zero coefficients are likely to be grouped in blocks rather than evenly distributed. Accordingly, N-BOMP is proposed to find a $(S_1, S_2, \ldots, S_N)$-block sparse representation of an N-mode tensor with respect to the factors $\mathbf{U}_n \in \mathbb{R}^{I_n \times M_n} (n = 1, 2, \ldots, N)$. If we denote by $\mathbf{B}_n \in \mathbb{R}^{I_n \times S_n}$ the submatrices formed by the columns indicated by indices $\mathbf{i}_n$ of the mode-n dictionary, i.e. $\mathbf{B}_n = \mathbf{U}_n(:, \mathbf{i}_n)$, then the approximation of the tensor can be written in the equivalent vector version of Eq. (8) in terms of Kronecker products of dictionaries, that is

$$\hat{\mathbf{y}} = (\mathbf{B}_N \otimes \mathbf{B}_{N-1} \otimes \cdots \otimes \mathbf{B}_1)\mathbf{x} \qquad (13)$$

where $\hat{\mathbf{y}} \in \mathbb{R}^{I_1 I_2 \ldots I_N}$ is the vectorized version of original tensor $\mathcal{X}$ by stacking all the columns of mode-1 tensor $\mathcal{Y}_{(1)}$ in a single vector, and $\mathbf{x} \in \mathbb{R}^{S_1 S_2 \ldots S_N}$ is the vectorized version of the N-mode tensor consisting only of non-zero entries. Here we assume that the size of the core tensor $\mathcal{X}$ is not less than the size of $\mathcal{Y}$ ($M_n \geq I_n$), because sparse coding model is formulated with overcomplete dictionaries. The N-BOMP algorithm is given in Algorithm 1.

Remark 1: N-BOMP not only optimizes the memory usage but also requires much less iterations ($K_{\max}$ against $S = S_1 S_2 \cdots S_N$ in the classical OMP/Kronecker-OMP algorithm), where S is the number of non-zero entries within the core tensor $\mathcal{X}$. Besides, the N-BOMP algrithm complexity in terms of the number of entries $I_1 I_2 \cdots I_N$, is sublinear compared to a linear dependence of the standard OMP and the Kronecker-OMP algorithms.

Algorithm 1 Solving Problem (11) by N-BOMP

Require: data tensor $\mathcal{Y}_m \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, mode-$n$ dictionaries$\{\mathbf{U}_i\} (1 \leq i \leq N)$ with $\mathbf{U}_i \in \mathbb{R}^{I_i \times M_i}$, the maximum number of non-zero entries K_{max}, error threshold ϵ

Ensure: Sparse representation $\mathcal{Y}_m = \mathcal{X}_m \times_1 \mathbf{U}_1 \times_2 \ldots \times_N \mathbf{U}_N$ with $x_{i_1 i_2 \cdots i_N} = 0 \; \forall (i_1, i_2, \ldots, i_N) \notin \mathbf{i}_1 \times \mathbf{i}_2 \times \cdots \times \mathbf{i}_N$ (with non-zero entries given by $\mathcal{X}_m(\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_N) = \mathcal{A}$)

$\mathbf{i}_n = [\emptyset] (n = 1, 2, \ldots, N)$, $\mathcal{R} = \mathcal{Y}_m$, $\mathcal{X}_m = 0$, $k = 1$;
while $|\mathbf{i}_1||\mathbf{i}_2| \cdots |\mathbf{i}_N| \leq K_{max}$ and $\|\mathcal{R}\|_F > \epsilon$ **do**
  $[i_1^k i_2^k \cdots i_N^k] = arg\max_{[i_1 i_2 \cdots i_N]} |\mathcal{R} \times_1 \mathbf{U}_1^T(:, i_1) \times_2 \cdots \times_N \mathbf{U}_N^T(:, i_N)|$;
  $\mathbf{i}_n = \mathbf{i}_n \cup [i_n^k] \; (n = 1, 2, \ldots, N)$, $\mathbf{B}_n = \mathbf{U}_n(:, \mathbf{i}_n)$;
  $\mathbf{a} = argmin_{\mathbf{w}} \|(\mathbf{B}_N \otimes \mathbf{B}_{N-1} \otimes \cdots \otimes \mathbf{B}_1)\mathbf{w} - \mathbf{y}\|_2^2$;
  $\mathcal{R} = \mathcal{Y}_m - \mathcal{A} \times_1 \mathbf{B}_1 \times_2 \mathbf{B}_2 \cdots \times_N \mathbf{B}_N$;
  $k = k + 1$;
end while
return $\{\mathbf{i}_1, \mathbf{i}_2, \ldots, \mathbf{i}_N\}, \mathcal{A}$;

B. Joint multiple dictionary update algorithm

Once the sparse core tensors $\mathcal{X}_m$ $(1 \leq m \leq M)$ are obtained, the tensor dictionaries are computed by solving the problem (12). Because non-zero atoms entries from each mode dictionary jointly make contributions to the presentation of tensors, we propose a joint multiple dictionary update algorithm based on the frequent atom patterns for sparse tensor representation.

1) Mining Frequent Atom Patterns in Sparse Representation: Each non-zero entry of the sparse tensor $\mathcal{X}_m (1 \leq m \leq M)$ is mapped into an integer using a function ϕ, which is formulated as

$$\begin{aligned} \phi(i_1 i_2 \ldots i_N) = {} & (i_1 - 1) I_2 I_3 \ldots I_N \\ & + (i_2 - 1) I_3 I_4 \ldots I_N \\ & + \cdots + i_N \end{aligned} \qquad (14)$$

Then the sparse representation of the N-mode tensor $\mathcal{X}_m (1 \leq m \leq M)$ is denoted by an integer set $\mathbf{t}_i$ consisting of indices of the non-zero entries of $\mathcal{X}_m$.

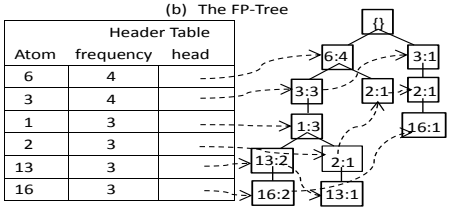
In order to uncover the atom correlations for sparse representation with a low time cost, we adopt FP-tree [8] to find the frequent patterns from M integer sets. The time efficiency of FP-Tree is achieved with three techniques: (1) FP-Tree avoids costly repeated data base scans by compressing a large database into a condensed smaller data structure; (2) a pattern-fragment growth method is employed to avoid the costly generation of a large number of candidate sets; and (3) it dramatically reduces the search space by decomposing the mining task into a set of smaller tasks for mining confined patterns in conditional database. Given a database

$\mathbf{T} = \{\mathbf{t}_1, \mathbf{t}_2, \ldots, \mathbf{t}_M\}$ and a minimum support threshold ξ, FP-Tree mines frequent patterns (i.e. the support of patterns is not less than ξ) by creating conditional (sub)pattern-bases (i.e., the subpattern-base under the condition of a frequent item's existence). Taking Figure 1 as an example, let the sparse representation database $\mathbf{T}$ be the first two columns of Fig. 1(a), and the minimum support threshold be 3 (i.e., $\xi = 3$). Firstly, a scan of $\mathbf{T}$ derives a list of frequent items in Fig. 1(b), (the number after ":" indicates the support), in which items are in frequency-descending order. Since each path of a tree will follow this order, the frequent atoms in each sparse representation are listed in the ordering in the rightmost column of Fig. 1(a). Secondly, the FP-Tree is constructed by scanning the database $\mathbf{T}$ again, together with the associated node-links pointing to the nodes with the same atom in the tree, as shown in Fig. 1(b). Finally, the frequent patterns related to each frequent atom are mined by traversing the FP-Tree once following corresponding node-links. The conditional pattern-base and conditional FP-Trees are generated during this process, as shown in Fig. 1(c). Hence, the frequent patterns related to a specific frequent atom are summarized in the second column of Fig. 1(c).

(a) An Atom database

SID	Atom Set	(Ordered) frequent Atoms
100	{6,1,3,4,7,9,13,16}	{6,3,1,13,16}
101	{1,2,3,6,12,13,15}	{6,3,1,,13}
102	{2,6,8,10,15}	{6,2}
103	{2,3,11,19,16}	{3,2,16}
104	{1,6,3,5,12,16,13,14}	{6,3,1,13,16}

(b) The FP-Tree



(c) Mining frequent patterns by creating conditional (sub)pattern-bases

Atom	Conditional pattern-base	Conditional FP-Tree
16	{(6,3,1,13:2),(3,2:1)}	{(3:3)}|16
13	{(6,3,1:2), (6,3,1,2:1)}	{(6:3,3:3, 1:3)}|13
2	{(6,3,1:1),(6:1),(3:1)}	{}
1	{(6,3:3)}	{(6:3, 3:3)}|1
3	{(6:3)}	{(6:3)}|3
6	{}	{}

Fig. 1: A toy database illustrates the FP-Tree frequent pattern mining process

2) Frequent Pattern based Dictionary Update: After finding frequent patterns related to each frequent atom, i.e., the conditional FP-Tree regarding a specific atom, we successively update all frequent patterns by jointly updating all the elements in each frequent pattern at a time. First of all, a map function φ is used to convert an integer into an N-dimensional vector, which is denoted by

$$\varphi(b) = i_1 i_2 \ldots i_N \tag{15}$$

where

$$\begin{aligned} m_1 &= \frac{b}{I_2 I_3 \ldots I_N} \\ i_1 &= m_1 + 1 \end{aligned} \tag{16}$$

$$i_{n+1} = \frac{\text{mod}(m_n)}{I_{n+2} \ldots I_N} + 1 (1 \le n \le N-2) \tag{17}$$

$$i_N = \text{mod}(\frac{\text{mod}(m_{N-1})}{I_N}) \tag{18}$$

Using Eq. (16)-(18), each element in the frequent pattern is converted into a position $(i_1, i_2, \ldots, i_N)$ of a sparse tensor $\mathcal{X}_m (1 \le m \le M)$. Then we define ω_k as the group of indices of sparse tensors $\mathcal{X}_m$ that use all the elements in the frequent pattern p_k. Thus

$$\omega_k = \{m | 1 \le m \le M, \mathcal{X}^m_{p_k} \ne 0\} \tag{19}$$

where $p_k = \{u_1^{k_1} u_2^{k_1} \cdots u_N^{k_1}, \ldots, u_1^{k_n} u_2^{k_n} \cdots u_N^{k_n}\}$, and $u_1^{k_j} u_2^{k_j} \cdots u_N^{k_j} (1 \le j \le n)$ is the mapped position of sparse tensor in the frequent pattern p_k.

We now turn to the second stage of updating the dictionaries together with the non-zero coefficients. Assume both $\mathcal{X}$ and $\mathbf{U}_i (1 \le i \le N)$, and we put in question only the dictionary atoms in each frequent pattern p_k, and the coefficients in $\mathcal{X}$ that corresponding to it. Let $\mathcal{X}_{p_k}$ and $\mathcal{Y}_{p_k}$ be an $(N+1)$-order tensor formed by stacking the subset ω_k of N-mode tensors $\mathcal{Y}_m$ and $\mathcal{X}_m$ along the $(N+1)$ mode, respectively; $\widetilde{\mathcal{X}}_{p_k}$ and $\widetilde{\mathcal{Y}}_{p_k}$ be the $(N+1)$-order tensor formed by N-mode tensors $\mathcal{Y}_m$ and $\mathcal{X}_m$ not in the subset ω_k . Then the penalty term of the objective function (12) can be rewritten as

$$\begin{aligned} &\|\mathcal{Y} - \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N \times_{N+1} \mathbf{I}\|_F^2 \\ =&\|\mathcal{Y} - \widetilde{\mathcal{X}}_{p_k} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N \times_{N+1} \mathbf{I} \\ &- \mathcal{X}_{p_k} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N \times_{N+1} \mathbf{I}\|_F^2 \\ =&\|\mathcal{E}_{p_k} - \mathcal{X}_{p_k} \times_1 \mathbf{U}_1 \times_2 \mathbf{U}_2 \cdots \times_N \mathbf{U}_N \times_{N+1} \mathbf{I}\|_F^2 \end{aligned} \tag{20}$$

where $\mathcal{E}_{p_k}$ stands for the error for all the $|\omega_k|$ examples when the dictionary atoms in the frequent pattern p_k are removed. We obtain $\mathcal{E}^R_{p_k}$ by choosing only the $(N+1)$-order indices in ω_k, as shown in Eq.(21).

$$\mathcal{E}^R_{p_k} = \mathcal{Y}_{(:,:,\ldots,:,\omega_{p_k})} - \mathcal{X}_{(i_1,i_2,\cdots,i_N,\omega_{p_k})} \times_1 \mathbf{U}_1 \cdots \times_N \mathbf{U}_N \times_{N+1} \mathbf{I} \tag{21}$$

where $(i_1, i_2, \cdots, i_N) \notin p_k$. We conduct a rank $(|p_k|, |p_k|, \ldots, |p_k|)$ CP decomposition for $\mathcal{E}^R_{p_k}$, in which the CP factors for mode $1, 2, \ldots, N$ of the j-th rank one tensor are regarded as the updated dictionary atoms of the j-th element in the frequent pattern p_k. The general process of joint multiple dictionary update algorithm is illustrated in Algorithm 2.

C. The complete Algorithm

After iteratively solving subproblem (11) and (12) until the maximum iterations are achieved or the iteration converges to stop, we finally obtain multiple overcomplete dictionaries along each mode of input tensors $\mathcal{Y}_m (1 \le m \le M)$ and corresponding sparse tensor representations $\mathcal{X}_m (1 \le m \le$

Algorithm 2 Solving Problem (12) by Joint Multiple Dictionary Update Algorithm

Require: data tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times M}$, sparse representation $\mathcal{X} \in \mathbb{R}^{M_1 \times M_2 \times \cdots \times M_N \times M}$ for $1 \leq i \leq M$, a minimum support threshold ξ

Ensure: Dictionaries from each mode $\mathcal{Y} = \mathcal{X} \times_1 \mathbf{U}_1 \times_2 \ldots \times_N \mathbf{U}_N$)

Initialize a non-zero entry database $\mathbf{T} = \{\}$
for $i = 1$ to M **do**
  An integer set for the i-th slice of $\mathcal{X}$: $n_{\mathcal{X}_m} \leftarrow$ non-zero entries in $\mathcal{X}_m$;
  $\mathbf{t}_i \leftarrow \{\}$
  for $j = 1$ to $|n_{\mathcal{X}_m}|$ **do**
    $\mathbf{t}_i \leftarrow \mathbf{t}_i \cup \phi(n_{\mathcal{X}_m}(j))$;
  end for
  $\mathbf{T} \leftarrow \mathbf{T} \cup \mathbf{t}_i$;
end for
a frequent pattern list $\mathbf{P} = \{p_1, p_2, \ldots, p_u\} \leftarrow$ Apply the FP-Tree algorithm to $\mathbf{T}$;
for $k = 1$ to u **do**
  for $j = 1$ to $|p_k|$ **do**
    $u_1^{k_j} u_2^{k_j} \cdots u_N^{k_j} \leftarrow \varphi(p_k(j))$;
    $p_k(j) \leftarrow u_1^{k_j} u_2^{k_j} \cdots u_N^{k_j}$
  end for
  $\omega_k \leftarrow \{i | 1 \leq i \leq M, \mathcal{X}_{p_k}^i \neq 0\}$;
  Calculate the representation error tensor $\mathcal{E}_{p_k}^R$ with Eq.(21);
  Do CP decomposition to the error tensor by a rank $(|p_k|, |p_k|, \ldots, |p_k|)$ approximation: $\mathcal{E}_{p_k}^R \leftarrow \sum_{i=1}^{p_k} \mathbf{a}_{p_k}^{i(1)} \circ \cdots \circ \mathbf{a}_{p_k}^{i(N)}$
  for $j = 1$ to $|p_k|$ **do**
    Update the dictionary atoms in the j-th element in p_k: $u_i^{k_j} = \mathbf{a}_{p_k}^{j(i)}$
  end for
end for
return $\{\mathbf{U}_1, \mathbf{U}_2, \ldots, \mathbf{U}_N\}$;

Algorithm 3 Joint multiple dictionary learning for tensor sparse coding

Require: data tensors $\mathcal{Y}_m \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} (1 \leq m \leq M)$, the maximum number of non-zero entries K_{max}, error threshold ϵ, a minimum support threshold ξ

Ensure: Sparse representation corresponding the learnt multiple dictionaries along each structure mode $\mathcal{Y}_m = \mathcal{X}_m \times_1 \mathbf{U}_1 \times_2 \ldots \times_N \mathbf{U}_N (1 \leq m \leq M)$)

Initialize dictionaries along each mode $\mathbf{U}_j (1 \leq j \leq N)$;
while reach maximum iteration times or converge to stop **do**
  Get the sparse representation tensors $\mathcal{X}_m (1 \leq m \leq M)$ by using N-BOMP;
  Update dictionaries $\mathbf{U}_j (1 \leq j \leq N)$ by using TSC-JMDL;
end while
return Sparse representation $\mathcal{X}_m (1 \leq m \leq M)$ and Dictionaries for each mode $\mathbf{U}_j (1 \leq j \leq N)$

M). Algorithm 3 outlines the whole process of joint multiple dictionary learning algorithm for tensor sparse coding.

IV. Experimental Results

In this section, we present a set of experimental results on both synthetic and real data sets with multi dimensional information. The intension of these experiments is to demonstrate our new method TSC-JMDL's superiority over the state-of art dictionary learning methods in computation complexity, memory usage, and image denoising.

A. Baseline Methods

As our proposed method is closely related to tensor dictionary models based on sparse TUCKER decomposition, we implement two multiple based dictionary learning algorithms for tensor sparse representation as baseline methods. All use the N-BOMP algorithm for sparse representation learning. However, they make use of two different dictionary update methods. These two baselines are listed as follows:

- **Tensor MOD Approach (TMOD)**: the original MOD algorithm is presented in [6] for dictionary learning on signal vectors. We apply this algorithm to tensor based multiple dictionary learning, which alternatively updates $\mathbf{U}_n$ by fixing $\mathbf{U}_1$, …,$\mathbf{U}_{n-1}$, $\mathbf{U}_{n+1}$,…, $\mathbf{U}_N$ to minimizing (12)
- **Tensor K-SVD Approach (TKSVD)**: Due to the atoms from each mode dictionary jointly make contributions to the tensor sparse representation, the K-SVD algorithm is unable to directly applied to tensor based dictionary learning by updating dictionary atom one by one. Instead, we check the coefficients $x_{i_1,i_2,\ldots,i_N}^m$ at each position $(i_1, i_2, \ldots, i_N)$ of all the sparse tensors $\mathcal{X}_m (1 \leq m \leq M)$. For those $x_{i_1,i_2,\ldots,i_N}^m \neq 0$, we gather the corresponding data tensor $\mathcal{X}_m$ to generate an $(N+1)$-order tensor $\mathcal{X}_s$, and conduct a rank-$(1, 1, \ldots, 1)$ CP decomposition. The CP factors for modes $1, 2, \ldots, N$ are regarded as the updated dictionary atoms.

We also consider some other sparse representation models proposed in [16], [5] and [17] to evaluate the performance in image denoising.

B. Synthetic Datasets

In this section, we evaluate TSC-JMDL against TMOD and TKSVD on the synthetic datasets. An $(N+1)$-order tensor of size $I_1 \times I_2 \ldots \times I_N \times 100 (I_n = 10$ for $n = 1, 2, \ldots, N$ is generated from the mode dictionaries of size $I_n \times M_n$ and the sparse core tensor of size $M_1 \times M_2 \ldots \times M_n \times 100$ whose elements are obtained from Gaussian distributions, where $M_n = 2I_n$. The sparse core tensor has a fixed mode sparsity of $S_n = 5$.

1) Time Cost with High Order Tensorial Data: To show TSC-JMDL's time advantage of handling high order tensorial data over other baseline methods, we create 6 higher order tensors $\mathcal{X}_j (2 \leq j \leq 7)$ using above tensor construction method. Fig. 2 reports the results on TMOD, TKSVD and our

new method TSC-JMDL. As the order of tensor increases, the time cost of TSC-JMDL and TMOD are significantly reduced compared with TKSVD. This is mainly because TKSVD jointly updates atoms, one for each mode at a time, the total number of these joint atoms is 20^{j-1}, which makes the dictionary update procedure particularly tedious. While TMOD simply updates each mode dictionary separately, the computation complexity of which is linear to the order of tensor. TSC-JMDL only jointly updates the frequent atom patterns in the sparse representation, which speeds up the dictionary update process. As TSC-JMDL needs extra time to find these frequent patterns, the time cost of TSC-JMDL is a little expensive than TMOD. To sum up, our method TSC-JMDL has a comparable time cost in the higher mode of tensor.

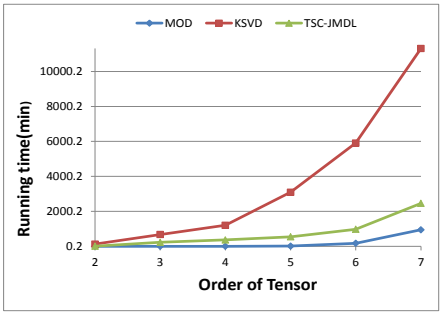


Fig. 2: Time comparison *w.r.t.* different orders of a tensor.

2) Data Denoising with Different Noise Parameters: We investigate the performance of TSC-JMDL for data denoising by comparing with TMOD and TKSVD. The test data $3-$order tensor is generated using above the same tensor construction method. The Gaussian noise is artificially added with zero mean. Fig. 3 presents the comparison of peak signal-to-noise ratios (PSNR) of the denoising results with increasing the noise parameter σ from 5 to 100. It can be found that joint multiple dictionary update algorithms TKSVD and TSC-JMDL outperform independent dictionary update algorithm TMOD, which suggests that considering atoms correlations for sparse representation can boost performance in denoising. Moreover, the performance of TKSVD is marginally better than the proposed method TSC-JMDL. This is mainly because TSC-JMDL updates merely frequent atoms patterns rather than all the non-zero coefficients in the sparse tensors. However, our new method TSC-JMDL can achieve a comparable performance with much lower time cost than TKSVD.

3) Memory Usage with respect to Dictionaries: Exemplified dictionaries used in a $3-$order tensor (created using above tensor generation method) is illustrated in Fig. 4. The left two images presents the dictionaries $\mathbf{U}_1$ and $\mathbf{U}_2$ learnt by our method TSC-JMDL, in which each column is the atom along one direction of the data. The right image is the Kronecker product $\mathbf{U}$ of above two dictionaries, which fully represents the data spatial correlation. Note that the size of dictionaries are quite different. The classical OMP

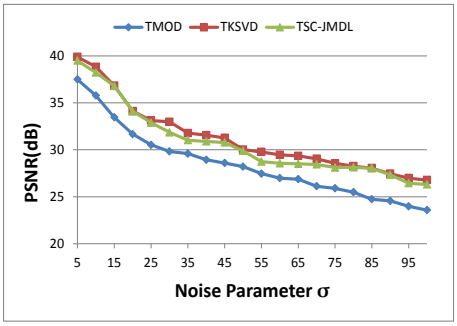


Fig. 3: The PSNR of denoising results with the variation of σ.

requires a large dictionary of size 100×400, whereas N-BOMP employed in our scheme needs only two dictionaries of size 10×20. Though only $1/100$ size of the Kronecker dictionary is used, our model can achieve better performance than the classical OMP algorithm. This is mainly because our model simultaneously selects factors along each mode on the original structured data, while 1D sparse model OMP chooses the factors on the reshaped $1D$ vector.

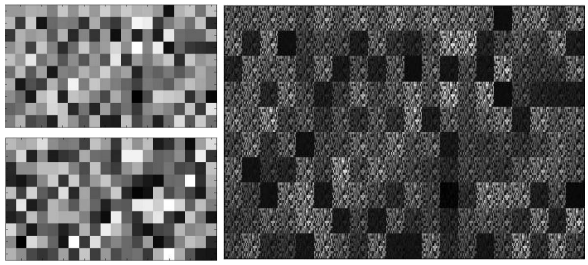


Fig. 4: Exemplified dictionaries in our 3-order tensor model.

C. Real Datasets

In this section, our proposed TSC-JMDL is used to solve the image denoising problem. The test images include 5 widely used images [5], commonly known as "Lena", "Peppers", "House", "Barbara", and "Boats", which are resized uniformly into 512×512. The white Gaussian noises are added at different standard deviations σ. In all the tests, image patch is of size 8×8 pixels, and the two dictionaries $\mathbf{U}_1$ and $\mathbf{U}_2$ are of size 8×16. Then $\mathbf{U}$ is generated by using Kronecker product is of size 64×256.

1) Image Denoising with Different Noise Variances: We study the performance of our TSC-JMDL in image denoising with those of $1D$ sparse coding models proposed in [16] and [5] and one $2D$ synthesis sparse model proposed in [17]. Table I shows the denoising results in terms of PSNR. We observe that our method outperforms the $2D$ synthesis sparse model [17] among all the data sets with different variations. This fact demonstrates that considering atom correlations for sparse representation can boost image

denoising performance. Note that the size of dictionaries are different in generating the results. The methods in [16] and [5] require the dictionaries of size 64×256, whereas our method only needs two dictionaries of size 8×16, which is $1/64$ size of the dictionary used in the above two $1D$ sparse coding models. However, our model TSC-JMDL can achieve the best performance among all the four evaluated methods as denoted by the bolded numbers.

TABLE I: Summary of the denoising PSNR results in [dB]. In each cell, the top row is the result of Portilla et al. [16], the second row is the result of Elad et al. [5]. They all use the dictionary of size 64×256. The third row is the result of Qi et al. [17], and the bottom row is the result of the proposed method. They all use two dictionaries of size 8×16.

$\sigma \setminus PSNR$	$2 \setminus 42.11$	$5 \setminus 34.15$	$10 \setminus 28.13$
Lena	42.23	38.49	35.61
	43.58	38.6	35.47
	43.58	38.55	35.37
	47.01	**42.12**	**39.96**
Barbara	43.29	37.79	34.03
	43.67	38.08	34.42
	43.64	38.05	34.02
	44.65	**39.77**	**36.37**
Peppers	43	37.31	33.77
	43.33	37.78	34.28
	43.37	37.93	34.26
	46.57	**41.98**	**37.02**
House	44.07	38.65	35.35
	44.47	39.37	35.98
	44.38	39.14	35.59
	48.98	**45.31**	**41.07**
Boats	42.09	36.97	33.58
	43.14	38.08	33.64
	43.11	37.16	33.56
	43.42	**38.01**	**34.39**

2) Performance with the Same Size of Dictionaries: We further study the performance of our method with the same size of dictionaries used in other baseline methods. Table II shows the denoising results of [17] and [5] using different sizes of dictionaries which are equal to the size used in our method. The noise parameter $\sigma = 5$, and the noise image of PSNR$= 34.15dB$. Clearly, the larger the size of dictionaries, the higher PSNR results all the methods achieve. Besides, two dictionary sparse models (TSC-JMDL and $2D$ synthesis sparse model [17]) significantly outperforms the $1D$ sparse model [5] when same size of dictionary is used. Another interesting fact is that the denoising result of our proposed method TSC-JMDL always performs better than the $2D$ synthesis sparse model, which again demonstrates that taking atom correlations into consideration can improve denoising results.

A visual comparison is given in Fig. 5. It presents the denoising results of Lena, Peppers and House generated by $1D$ sparse model [5], $2D$ synthesis sparse model [17] and our proposal model TSC-JMDL with the dictionary of the same sizes 64×4 and $2 \times 8 \times 16 = 256$, respectively. Obviously, our method TSC-JMDL provides much clearer reconstructed image than our two baseline methods.

TABLE II: Summary of the denoising PSNR results in [dB]. In each cell, the top row is the result of Elad et al. [5]. It uses the dictionaries of size 64×4, 64×16 and 64×64 respectively. The middle row is the result of Qi et al. [17], and the bottom row is the result of the proposed method. They all use two dictionaries of size 8×16, 8×64 and 8×256 respectively.

$image \setminus DictSize(pixels)$	256	1024	4096
Lena	30.09	35.81	38.20
	38.55	44.86	47.12
	42.12	**47.25**	**49.94**
Barbara	24.40	30.14	37.74
	38.05	41.36	46.76
	39.77	**43.02**	**49.96**
Peppers	25.73	32.23	37.29
	37.93	42.28	45.16
	41.98	**45.98**	**48.79**
House	29.35	36.15	39.34
	39.14	44.68	46.59
	45.31	**47.99**	**49.27**
Boats	27.08	33.04	37.14
	37.16	41.78	48.09
	38.01	**43.03**	**49.78**

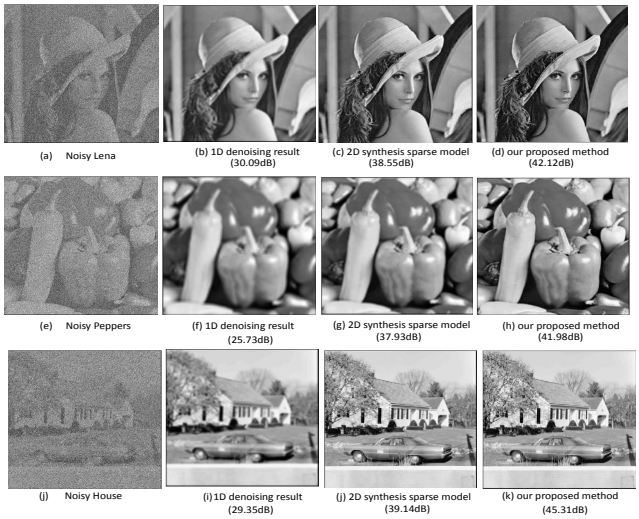
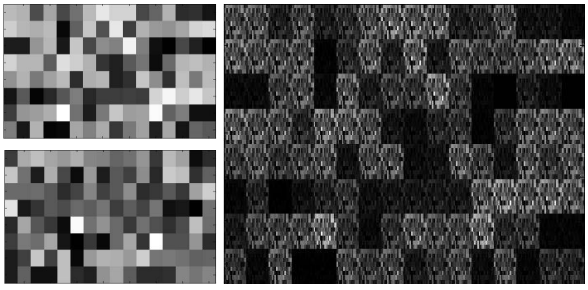


Fig. 5: The denoising results of Lena, Peppers and House by $1D$ sparse model [5], $2D$ synthesis sparse model [17] and our proposal model TSC-JMDL using the dictionaries of size 64×4 and 8×16, respectively.

Exemplified dictionaries of Lena used in our method is illustrated in Fig. 6. The $2D$ dictionaries $\mathbf{U}_1$, $\mathbf{U}_2$ denote different dimensional features, and the Kronecker product $\mathbf{U}$ fully represents the image spatial correlations along each direction. Two dictionaries in our model are learnt by exploring the structure information along each mode and their correlations among all the possible modes, whereas the Kronecker product $\mathbf{U}$ is generated depending on one direction information on the reshaped image vector. Thus, our model always has better image denosing results than $1D$ sparse model.



(a) Our Dictionaries $\mathbf{U}_1, \mathbf{U}_2$ (b) The Kronecker product $\mathbf{U}$

Fig. 6: Exemplified dictionaries of Lena in our model TSC-JMDL. Left two images show $\mathbf{U}_1$, $\mathbf{U}_2$, in which each column is the atom of one directional signal of the image patch. The right image is the Kronecker dictionary $\mathbf{U}$ in which each square is an atom of size 8×8.

V. Conclusion

In this paper, we propose a novel multiple dictionary learning algorithm for tensor sparse coding. While other existing tensor dictionary learning algorithms update each mode dictionary by fixing all the other mode dictionaries iteratively in an alternating manner, the proposed method fully makes use of the atom correlations among all the spatial modes inside a higher order tensorial data. Our model TSC-JMDL employs the FP-tree mining algorithm to exploit frequent atom patterns in the sparse representation, and then simultaneously updates multiple atoms in the frequent pattern using our joint dictionary update method. On the synthetic datasets, we show that our model can achieve a comparable performance with a lower computation gain and memory usage. Moreover, we also demonstrate its effectiveness in data denoising with different noise variances. On the real-world datasets, our method shows promising results in image denoising. Our model outperforms both tensor dictionary learning methods and traditional $1D$ models, with different noise parameters and similar memory cost. The image reconstruction results clearly show the ability of our algorithm for maintaining the discerning features while retaining the image reconstruction.

References

[1] M. Aharon, M. Elad, and A. Brucktein. K-SVD: An algorithm for desdesign overcomplete dictionaries for sparse representation. *IEEE Trans. on Signal Processing*, 54(2):4311–4322, 2006.

[2] E. Benetos and C. Kotropoulos. Non-negative tensor factorization applied to music genre classification. *Audio, Speech, and Language Processing, IEEE Transactions on*, 18(8):1955–1967, 2010.

[3] M. Blondel, K. Seki, and K. Uehara. Block coordinate descent algorithms for large-scale sparse multiclass classification. *Machine Learning*, 93(1):31–52, 2013.

[4] C. F. Caiafa and A. Cichocki. Computing sparse representations of multidimensional signals using kronecker bases. *Neural Comput.*, 25(1):186–220, January 2013.

[5] M. Elad, M.and AElad. Image denoising via sparse and redundant representation over learned dictionaries. *Image Processing, IEEE Transactions on*, 15:3736–3745, 2006.

[6] K. Engan, S. Aase, and J.H. Husoy. Method of optimal directions for frame design. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 2443–2446, 1999.

[7] Y. Fang, J.J. Wu, and B.M. Huang. 2d sparse signal recovey via 2d orthogonal matching pursuit. *Science China Information Sciences*, 55:889–897, 2012.

[8] J. Han, J. Pei, Y. Yin, and R. Mao. Mining frequent patterns witwith candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8:53–87, 2004.

[9] T. Hazan, S. Polak, and A. Shashua. Sparse image coding using a 3d non-negative tensor factorization. In *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, volume 1, pages 50–57 Vol. 1, 2005.

[10] Y. Kim and S. Choi. Nonnegative tucker decomposition. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1–8, 2007.

[11] Y. Kim, A. Cichocki, and S. Choi. Nonnegative tucker decomposition with alpha-divergence. In *ICASSP*, pages 1829–1832. IEEE, 2008.

[12] G. Kolda and B. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[13] S. Li. Non-negative sparse coding shrinkage for image denoising using normal inverse gaussian density model. *Image Vision Comput.*, 26(8):1137–1147, August 2008.

[14] M. Mørup, L.K. Hansen, and S. M. Arnfred. Algorithms for sparse nonnegative tucker decompositions. *Neural Comput.*, 20(8):2112–2131, August 2008.

[15] G. Peyré. Sparse Modeling of Textures. *Journal of Mathematical Imaging and Vision*, 34(1):17–31, May 2009.

[16] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using scale mixtures of gaussian in the wavelet domain. *Image Processing, IEEE Transactions on*, 12:1338–1351, 2003.

[17] N. Qi, Y. Shi, X. Sun, J. Wang, and B. Yin. Two dimensional synthesis sparse model. In *Multimedia and Expo (ICME), 2013 IEEE International Conference on*, pages 1–6, 2013.

[18] J. Yang, K. Yu, Y. Gong, and T. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *in IEEE Conference on Computer Vision and Pattern Recognition(CVPR)*, 2009.

[19] S. Zubair and W. Wang. Tensor dictionary learning with sparse tucker decomposition. In *Digital Signal Processing (DSP), 2013 18th International Conference on*, pages 1–6, 2013.